



Міністерство освіти і науки України  
Національний технічний університет України  
“Київський політехнічний інститут імені Ігоря Сікорського”  
Факультет інформатики та обчислювальної техніки  
Кафедра автоматики та управління в технічних системах

**Лабораторна робота №4**  
**Архітектура комп'ютера**  
«ДОСЛІДЖЕННЯ РОБОТИ СТЕКУ МПС В АРХІТЕКТУРІ ІА-32 (X86) У  
REAL ADDRESS MODE»

Виконали:  
студенти групи ІТ-01:

Перевірив:  
Бердник Ю. М.

Бардін В.Д.  
Задніпрянець А.А.  
Куркін О.О.

Дата здачі 08.04.2021

Захищено з балом \_\_\_\_\_

Київ 2021

## Вихідний код:

```
1 TITLE ЛР_4
2 ;
3 ;ЛР №4
4 ;
5 ; Комп'ютерна архітектура
6 ; ВУЗ: НТУУ "КПІ"
7 ; Факультет: ФІОТ
8 ; Курс: 1
9 ; Група: IT-01
10 ;
11 ; Автор: Бардін В. Д.
12 ; Задніпрянець А. А.
13 ; Куркін О. О.
14 ;
15 ; Дата: 04/04/21
16 ;
17 IDEAL ; Директива - тип Асемблера tasm
18 MODEL small ; Директива - тип моделі пам'яті
19 STACK 1024 ; Директива - розмір стеку
20
21 DATASEG
22 array_stack dw 2E2Eh, 2098h, 6847h, 7230h, 5182h, 1964h, 7350h, 1628h, 9823h, 5927h, 2343h, 7515h, 5238h, 8272h, 6576h, 2619h
23 dw 9520h, 6608h, 5535h, 9745h, 8314h, 1683h, 2065h, 3718h, 3198h, 5267h, 1546h, 2273h, 9317h, 2181h, 7466h, 1940h
24 dw 7417h, 2385h, 2240h, 8346h, 9246h, 9573h, 5050h, 1549h, 4402h, 7854h, 8126h, 9060h, 3476h, 7497h, 3703h, 1857h
25 dw 8794h, 8017h, 3227h, 1033h, 7980h, 8658h, 6475h, 2653h, 4970h, 3343h, 3788h, 4600h, 4953h, 5156h, 7128h, 9539h
26 dw 4889h, 8734h, 8685h, 3104h, 5514h, 9721h, 5958h, 4611h, 7759h, 1725h, 6059h, 7499h, 9681h, 4573h, 6929h, 1387h
27 dw 2772h, 5484h, 7392h, 8250h, 5503h, 6000h, 1249h, 3413h, 3167h, 9250h, 7496h, 7533h, 2101h, 4007h, 6810h, 4531h
28 dw 4979h, 9050h, 8589h, 6962h, 5374h, 3451h, 2971h, 6612h, 8002h, 4074h, 2634h, 3694h, 7979h, 7571h, 7333h, 9852h
29 dw 9847h, 3270h, 242Ch, 6722h, 2697h, 9765h, 2203h, 3222h, 6819h, 5024h, 5846h, 3619h, 9626h, 2638h, 4465h, 2E2Eh
30
31 first_birthdate db "8.10|09.10|24.04"
32
33 CODESEG
34 Start:
35 mov ax, @data ; data segment init
36 mov ds, ax
37 mov es, ax
38
39 ; set params for copy_array
40 mov cx, 128 ; repeats amount
41 call copy_arr
42
43 ; set params for array_to_stack
44 mov cx, 128 ; repeats amount
45 call array_to_stack
46
47 ; set params for set_bdates
48 mov cx, 16 ; repeats amount
49 mov bp, 0150h ; offset for birthdates
50 call set_bdate
51
52 ; application finishing
53 mov ah, 4ch
54 int 21h
55
56 ;-----Copy array procedure-----
57 ; Input params: cx - initial array size,
58 ; Output params: array copy placed at ds,
59 ;
60 PROC copy_arr
61 xor si, si ; set si to zero
62 array_coping_loop:
63 mov bx, [ds:si] ; get number from array_array stack & set it to bx as a temp variable
64 mov [ds:si+260h], bx ; set value from bx to ds with offset
65 add si, 2 ; si value + 2
66 loop array_coping_loop
67
68 ret
69 ENDP
70
71 ;-----Put array to stack procedure-----
72 ; Input params: cx - initial array size,
73 ; Output params: array copy placed at ds,
74 ;
75 PROC array_to_stack
76 xor si, si ; set si to zero
77 push_to_stack:
78 mov ax, [ds:si] ; set value to ax from ds with offset si
79 push ax ; push ax value to stack
80 add si, 2 ; si + 2
81 loop push_to_stack
82
83 ret
84 ENDP
85
86 ;-----Add birthdate to stack-----
87 ; Вхідні параметри: cx - symbols amount at birthday,
88 ; bp - offset
89 ; Вихідні параметри: array copy placed at ds,
90 ; array begin address at ax
91 ;
92 PROC set_bdates
93 xor si, si ; set si to zero
94 birthdate_label:
95 mov ah, [first_birthdate+si] ; set value to ah from first_birthdate with offset si
96 mov [bp], ah ; add value to stack
97 inc si ; increment si
98 inc bp ; increment bp
99 loop birthdate_label
100
101 ret
102 ENDP
103 end Start
```

**Результати дослідження:** Було створено двовимірний масив, що складається з елементів у два байти (слово) кожен, має розмір 16x8 (для зручності виведення). Потім за допомогою процедури `copy_arr` було зроблено копію масиву.

```
;-----Copy array procedure-----
; Input params: cx - initial array size,
; Output params: array copy placed at ds,
;-----
PROC copy_arr
    xor si, si                ; set si to zero
    array_coping_loop:
        mov bx, [ds:si]      ; get number from array & set it to bx
        mov [ds:[si+260h]], bx ; set value from bx to ds with offset
        add si, 2            ; si value + 2
        loop array_coping_loop

    ret
ENDP
```

Далі за допомогою процедури `array_to_stack` помістили копію і стек.

```
;-----Put array to stack procedure-----
; Input params: cx - initial array size,
; Output params: array copy placed at ds,
;-----
PROC array_to_stack
    xor si, si                ; set si to zero
    push_to_stack:
        mov ax, [ds:[si]]    ; set value to ax from ds with offset si
        push ax              ; push ax value to stack
        add si, 2            ; si + 2
        loop push_to_stack

    ret
ENDP
```

А потім змінили значення в стеці використавши `set_bdates`.

```

;-----Add birthdate to stack-----
; Input params: cx - symbols amount at birthday,
;               bp - offset
; Output params: array copy placed at ds,
;-----
    PROC set_bdates
        xor si,si                ; set si to zero
        birthdate_label:
            mov ah, [first_birthday+si]
            mov [bp], ah          ; add value to stack
            inc si                ; increment si
            inc bp                ; increment bp
            loop birthdate_label

        ret
    ENDP

```

```
DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: TD
File Edit View Run Breakpoints Data Options Window Help READY
[ ]=CPU 80486-5[ ]
ss:0000 C505 lds ax,[di] ax 3402 c=0
ss:0002 0000 add [bx+si],al bx 6848 z=0
ss:0004 0000 add [bx+si],al cx 0000 s=0
ss:0006 0000 add [bx+si],al dx FF7C o=0
ss:0008 0000 add [bx+si],al si 0010 p=1
ss:000A 0000 add [bx+si],al di 0000 a=1
ss:000C 0000 add [bx+si],al bp 0160 i=1
ss:000E 0000 add [bx+si],al sp 0004 d=0
ss:0010 0000 add [bx+si],al ds 0881
ss:0012 0000 add [bx+si],al es 2098
ss:0014 0000 add [bx+si],al ss 0892
ss:0016 0000 add [bx+si],al cs 087C
ss:0018 0000 add [bx+si],al ip 001C
ss:001A 0000 add [bx+si],al
ss:001C 0000 add [bx+si],al

ss:014C 00 00 00 00 38 2E 31 30 8.10
ss:0154 7C 30 39 2E 31 30 7C 32 109.1012
ss:015C 34 2E 30 34 23 98 27 59 4.04#ij'Y
ss:0164 43 23 15 75 38 52 72 82 C#Su8Rré
ss:016C 76 65 19 26 20 95 08 66 ve1& ò.f

ss:014C 0000
ss:014A 0000
ss:0148 0000
ss:0146 0000
ss:0144 0000

Ctrl: G-Goto S-Search N-Next C-Change F-Follow P-Previous D-Display B-Block
```

```
DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: TD
File Edit View Run Breakpoints Data Options Window Help READY
[ ]=Dump-4[ ]
ds:0240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0260 38 2E 31 30 7C 30 39 2E 31 30 7C 32 34 2E 30 34 8.10109.10124.04
ds:0270 23 98 27 59 43 23 15 75 38 52 72 82 76 65 19 26 #ij'YC#Su8Rréve1&
ds:0280 20 95 08 66 35 55 45 97 14 83 83 16 65 20 18 37 ò.f5UEùMââ.e 17
ds:0290 98 31 67 52 46 15 73 22 17 93 81 21 66 74 40 19 ij1grFSS"tôüfttô↓
ds:02A0 17 74 85 23 40 22 46 83 46 92 73 95 50 50 49 15 tta#e"FaFfsdPPI§
ds:02B0 02 44 54 78 26 81 60 90 76 34 97 74 03 37 57 18 EDTx&iü`Év4ùt7W†
ds:02C0 94 87 17 80 27 32 33 10 80 79 58 86 75 64 53 26 öçtG'23>CyXâudS&
ds:02D0 70 49 43 33 88 37 00 46 53 49 56 51 28 71 39 95 pIC3ê7 FSIUQ(q9d
ds:02E0 89 48 34 87 85 86 04 31 14 55 21 97 58 59 11 46 ëH4çââ+1WU!ùXY4F
ds:02F0 59 77 25 17 59 60 99 74 81 96 73 45 29 69 87 13 YwçtY`ÖtiûsE)iç!!
ds:0300 72 27 84 54 92 73 50 82 03 55 80 60 49 12 13 34 r'âTfsPéUÇ`I†!!4
ds:0310 67 31 50 92 96 74 33 75 01 21 07 40 10 68 31 45 g1Pffût3u@!•e>h1E
ds:0320 79 49 50 90 89 85 62 69 74 53 51 34 71 29 12 66 yIPÉèàbitSQ4q)†f
ds:0330 02 80 74 40 34 26 94 36 79 79 71 75 33 73 52 98 BÇt@4&ô6yyqu3sRij
ds:0340 47 98 70 32 26 24 22 67 97 26 65 97 03 22 22 32 Gyp2&$"gù&èù"'"2
ds:0350 19 68 24 50 46 58 19 36 26 96 38 26 65 44 2E 2E 1h$PFX16&û8&eD..
ds:0360 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0370 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
ds:0380 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Ctrl: G-Goto S-Search N-Next C-Change F-Follow P-Previous D-Display B-Block
```

DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

Module: 14 File: 14.asm 50

• mov ax, @stack

• ; set params for set

• mov cx, 16

• mov bp, 0150h

• call set\_bdates

• ; application finish

• mov ah, 4ch

• int 21h

• ;-----

• ; Input params: cx -

• ; Output params: arr

• ;-----

• PROC copy\_arr

• xor si, si ; set si to zero

• array\_coping\_loop:

Registers:

ax	0881	c=0
bx	6848	z=1
cx	0000	s=0
dx	FF7C	o=0
si	4465	p=1
di	0000	a=0
bp	2E2E	i=1
sp	0004	d=0
ds	0881	
es	2098	
ss	0892	
cs	087C	
ip	0016	

Watches

Ctrl: I-Increment D-Decrement Z-Zero C-Change R-Registers

DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: TD

File Edit View Run Breakpoints Data Options Window Help

Dump

ss:0000 C5 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ss:0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ss:0020 2E 2E 98 20 47 68 30 72 82 51 64 19 50 73 28 16 ..ij Gh0réQd1Ps(.

ss:0030 23 98 27 59 43 23 15 75 38 52 72 82 76 65 19 26 #ij'YC#Su8Rréve1&

ss:0040 20 95 08 66 35 55 45 97 14 83 83 16 65 20 18 37 òf5UEù1lââ.e 17

ss:0050 98 31 67 52 46 15 73 22 17 93 81 21 66 74 40 19 ü1gRF8s"tôü!ftø!

ss:0060 17 74 85 23 40 22 46 83 46 92 73 95 50 50 49 15 tta#e"FaFfsøPPI8

ss:0070 02 44 54 78 26 81 60 90 76 34 97 74 03 37 57 18 ØDTx&ü`Év4ùt♥7Wt

ss:0080 94 87 17 80 27 32 33 10 80 79 58 86 75 64 53 26 öc±Q'23>CyXäudS&

ss:0090 70 49 43 33 88 37 00 46 53 49 56 51 28 71 39 95 pIC3ê7 FSIUQ(q9ð

ss:00A0 89 48 34 87 85 86 04 31 14 55 21 97 58 59 11 46 ëH4gââ•1WU!ùXY4F

ss:00B0 59 77 25 17 59 60 99 74 81 96 73 45 29 69 87 13 Ywz:±Y'ÖtüûsE)ig!!

ss:00C0 72 27 84 54 92 73 50 82 03 55 80 60 49 12 13 34 r'äTHsPé♥UQ`I!#4

ss:00D0 67 31 50 92 96 74 33 75 01 21 07 40 10 68 31 45 g1Pâût3u@!•@h1E

ss:00E0 79 49 50 90 89 85 62 69 74 53 51 34 71 29 12 66 y

ss:00F0 02 80 74 40 34 26 94 36 79 79 71 75 33 73 52 98 8

ss:0100 47 98 70 32 26 24 22 67 97 26 65 97 03 22 22 32 6

ss:0110 19 68 24 50 46 58 19 36 26 96 38 26 65 44 2E 2E ↓

ss:0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ss:0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ss:0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Registers:

ax	0881	c=0
bx	6848	z=1
cx	0000	s=0
dx	FF7C	o=0
si	4465	p=1
di	0000	a=0

Ctrl: I-Increment D-Decrement Z-Zero C-Change R-Registers

Скориставшись командою @stack ми знайшли логічне зміщення регістру. його адреса 0881. Далі знаючи це можна розрахувати логічні та фізичні адреси 1 і останнього елемента. 1 елемент має зміщення 0020 відносно

початку стеку, а отже логічна адреса дорівнює  $ss:0020$ . а фізична —  $(0881 * 16) + [ss] = 08810 + 0020 = 8830$ . Для останнього: логічна адреса =  $011Fh$ , а фізична —  $0892Fh$ .

**Висновок:** У ході лабораторної роботи попрацювали з різними видами адресації, створили двовимірний масив і розібралися як працює найшвидший спосіб роботи з даними: стек. Також на реальному прикладі побачили як виглядає код на Асемблері написаний у процедурному стилі і які переваги і недоліки він має у порівнянні з іншими стилями написання.